УДК 004.62

Фісун М. Т.,
д-р техн. наук, професор, завідувач кафедри ІІС, Чорноморський державний університет імені Петра Могили, м. Миколаїв, Україна
Стешов І. В.,
аспірант кафедри ІІС, Чорноморський державний університет імені Петра Могили, м. Миколаїв, Україна

РОЗРОБКА ПАРСЕРА ЛІТЕРАТУРНИХ ДЖЕРЕЛ ЗАСОБАМИ ANTLR ТА HIBERNATE ДЛЯ ФОРМУВАННЯ БІБЛІОГРАФІЧНОЇ БАЗИ ДАНИХ

Представлено реалізацію парсера бібліографічних описів літературних джерел у текстовому вигляді, що знаходить області опису та заносить їх у відповідні таблиці бази даних. Програмне забезпечення розроблено засобами генератора парсерів ANTLR. Наводяться лексичні і синтаксичні правила для реалізації парсера літературних джерел та інтерфейс програмного забезпечення.

Ключові слова: *бібліографічний опис, літературний ресурс, база даних, HSQLDB, Hibernate, Spring Data, ANTLR, парсер, лексер.*

Вступ. Відомо, що кожна стаття чи публікація має список джерел, на які спирається автор. Існування системи, що розпізнаватиме літературні джерела може допомогти у подальшому ознайомленні з темою, перевірці неоднозначних тез (зокрема, якщо є підозра у вандалізмі), мінімізації суперечок між авторами, уникнень звинувачень у плагіаті та крадіжок інтелектуальної власності.

Розробка парсера літературних джерел для формування бази даних можна виконати як шляхом розробки «власного» програмного забезпечення (лексичний і синтаксичний аналізатори), використовуючи складні регулярні вирази та/або формальні граматики, так і за допомогою інструментальних засобів побудови компіляторів, таких як JavaCC, BIZON, YACC, ZUBR [1], ANTLR [2] т. і. В результаті проведеного аналізу було обрано генератор парсерів ANTLR, так як автори вже мають досвіт у використанні саме цього інструменту[3].

Постановка задачі. Основна ідея проекту полягає у створенні програми, яка зможе розпізнавати бібліографічні описи літературних ресурсів та заповнювати ними базу даних.

Основні *проблеми*, що потребують вирішення в процесі роботи, пов'язані зі створенням структури бази даних, відображення таблиць з бази даних на Java класи, створення правил для парсінгу списку літературних джерел. Існує різні види можливих бібліографічних описів: наукові статті, книги, словники, атласи, підручники, каталоги, електронні ресурси, законодавчі матеріали, стандарти та ін. Враховуючи те, що розпізнають 8 основних областей опису [4; 5], де не всі області є обов'язковими та кожна область може поділятися на декілька підобластей, то множина можливих структур бібліографічних описів велика і час на створення універсального програмного забезпечення (ПЗ), яке б змогло розпізнати будь-який опис, потребує суттєвих витрат. Тому однією проблемою буде вибір підмножини правил, яке буде підтримувати система, що розроблюється.

Правила складання бібліографічних описів у переліку посилань. Бібліографічні описи літературних джерел виконуються відповідно до національного стандарту ДСТУ ГОСТ 7.1:2006 [5], який набув чинності 1 липня 2007 р. на заміну ГОСТ 7.1–84 та є міждержавним стандартом для 10 країн. Новий ДСТУ ГОСТ 7.1:2006 покликаний забезпечити впровадження сучасних автоматизованих технологій опрацювання документів, ведення інформаційних баз даних; ефективність пошуку та використання документів всіх видів та типів.

Бібліографічні описи в переліку посилань подають у порядку, за яким вони вперше згадуються в тексті. Порядкові номери описів в переліку є посиланнями в тексті (номерні посилання у квадратних дужках).

До бібліографічного опису входять такі області:

1) заголовку та відомостей про відповідальність;
2) видання;
3) специфічних відомостей;
4) вихідних даних;
5) фізичної характеристики;
6) серії;
7) приміток;
8) стандартного номера (чи його альтернативи) та умов доступності.

Області опису складаються з елементів, котрі підрозділяються на обов'язкові та факультативні. В описі можуть бути лише обов'язкові елементи або обов'язкові та факультативні (необов'язкові).

Структура бази даних представлена на рис. 1. Вона не претендує на досконалість і може бути змінена. На

цьому етапі досліджень автори доводять принципову можливість парсінгу текстового представлення бібліографічних описів та наповнення бази даних. Для відображення джерел достатньо було б усього однієї таблиці, де повна назва ресурсу була б унікальною (ключем), але для того щоб мати гнучкість та зручність у використанні програми, було прийнято рішення виділяти авторів та видавництва у окремі таблиці.

Рис. 1. Структура бази даних літературних джерел

Ця особливість зробить можливим перегляд статей у розрізі авторів та видавництв. Таким чином, якщо користувач забуде назву ресурсу, але, буде пам'ятати автора, він легко зможе знайти публікацію, просто змінивши відображення ресурсів на відображені у розрізі авторів. Таблиця «Автор» має унікальний ключ, прізвище автора та посилання на список його публікацій. Таблиця «Публікація» має унікальний ключ, список авторів, що працювали над нею, назву, позначення, місто публікації, видання, рік видання, кількість сторінок, серію та випуск. Таблиця «Видавництво» має унікальний ключ, назву, місто в якому видавництво розташовано, рік його створення, та список публікацій, що були видані за участю цього видавництва. Інші бібліографічні описи, які можуть зустрітися, поки що проігноровані, але будуть додані у наступних версіях програми.

Технології, що були використанні під час створення проекту:

1. Java 8 – основна мова програмування ;
2. Spring Framework 4 – Inversion of Control контейнер, контроль за транзакціями до бд, тестування;
3. HSQLDB – база даних;
4. Hibernate 4 – JPA implementation, ORM технологія для відображення таблиць бази даних у вигляді Java класів;
5. ANTLR 4 – генератор парсерів;
6. JavaFX – основний інструмент для графічного інтерфейсу;
7. Tiwulfx – бібліотека для графічного інтерфейсу, що має спеціальний компонент для представлення тих класів, що відображають таблиці з бази даних.
8. Log4j – бібліотека для логування;
9. Maven – технологія для зборки проекту та завантаження залежностей бібліотек з глобальних репозиторіїв;
10. Junit – бібліотека для проведення тестування.

Структура парсера літературних джерел представлена на рис. 2, з якого видно, що для автоматизації розробки лексичного аналізатору було використано одну з найбільш популярних систем – ANTLR, вхідною мовою якого є регулярні вирази [2]. ANTLR, як і багато інших подібних засобів, складається з бібліотеки класів, що полегшують основні операції при розборі (буферизація, пошук і т. д.), і утиліти, які генерують код парсера на основі файлу, що описує граматику мови, що розбирається. Сам ANTLR написаний на Java [2], але дозволяє генерувати код як на Java, так і на мовах C++, C#, JavaScrip та ще деяких популярних мовах.

Програма побудована на платформі Spring Framework, що дозволяє контролювати життєвий цикл усіх об'єктів та має гнучку конфігурацію усього додатку і, також, має розгорнуту конфігурацію, що налаштовує взаємодію з базою даних.

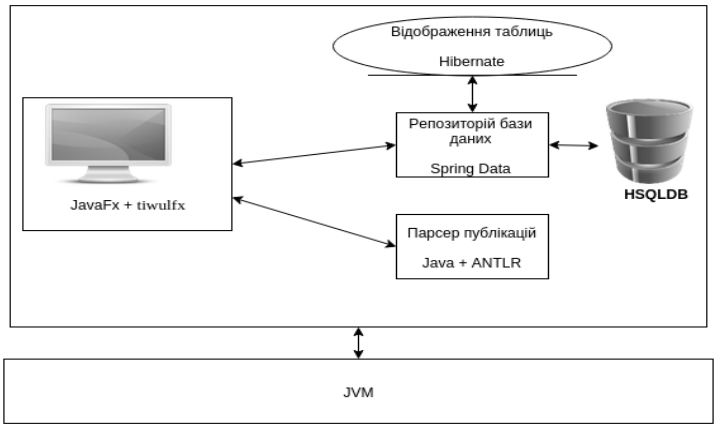

Рис. 2. Загальна структура парсера літературних джерел

Особливістю фреймворка Hibernate є реалізація відображення зі зв'язками таблиць з бази даних у Java й об'єкти (ORM – Object-relational Mapping). Такі Java класи називаються Entity, і, зазвичай, такі класи не

повинні містити ніякою логіки, вони повинні тільки описувати об'єкт, тобто бути тим, що називається POJO (Plain Old Java Object) або Java Bean. Spring конфігурація налаштована таким чином, що база даних створюється по вже існуючим Entity, а не навпаки.

Spring Data є основним шаром, що зв'язує Entity з базою даних, його особливість полягає у економії часу та коду при написанні запитів. Більшість загальних запитів до бази вже реалізовані у цій бібліотеці, а для того, щоб додати новий простий запит, наприклад вибірка по полю, потрібно просто описати сигнатуру метода, дотримуючись конвенції, у інтерфейсі без її реалізації.

Лексичний аналіз. Основна задача лексичного аналізу – розбити вхідний текст, що складається з послідовності одиночних символів, на послідовність слів чи лексем, тобто виділити ці слова з безперервної послідовності символів [6]. Усі символи вхідної послідовності з цього погляду розділяються на символи, що належать яким-небудь лексемам, і символи, що поділяють лексеми (роздільники).

Як джерело даних використовується клас потоку, що дозволяє розбирати дані, що знаходяться у файлі, в пам'яті процесу або приходять через мережу [2]. Результатом роботи ANTLR є два класи – лексер і парсер. Лексер розбиває потік символів на потік токенів відповідно до правил, а парсер обробляє потік токенів у відповідності з іншими правилами. ANTLR відноситься до так званих LL(*)-аналізаторів – не обмежених скінченним числом лексем для попереднього перегляду, а здатних приймати рішення, визначаючи, чи належать вхідні лексеми регулярній мові. Основні її правила наведено нижче в табл. 1.

Таблиця 1

Перелік правил регулярної мови граматики

№	Правило	Опис дії правила
1	(...)	підправило
2	(...)*	повтор підправила 0 чи більше разів
3	(...)+	повтор підправила 1 чи більше разів
4	(...)?	підправило може бути відсутнім
5	{...}	семантичні дії (на мові, яка використовується в якості вихідної, наприклад. Java)
6	[...]	параметри правила
7	|	оператор альтернативи
8	..	оператор діапазона
9	~	заперечення
10	.	будь-який символ

№	Правило	Опис дії правила
11	=	присвоювання
12	:	мітка, початок правила
13	;	кінець правила
14	class	клас граматики
15	extends	визначає базовий клас для граматики
16	returns	опис повертаємого значення для правила
17	options	секція опцій
18	tokens	секція токенів
19	header	заголовок

Лексичний аналізатор, генерований ANTLR, взаємо-ію з синтаксичним аналізатором таким чином: при виклику його синтаксичним аналізатором лексичний аналізатор по символьно читає залишок входу [7], поки не знаходить найдовший префікс, що може бути співставлений одному з регулярних виразів. Потім він виконує відповідну дію. Як правило, дія повертає керування синтаксичному аналізатору. Якщо це не так, тобто у відповідній дії немає повернення, то лексичний аналізатор продовжує пошук лексем до тих пір, поки дія не поверне керування синтаксичному аналізатору. Повторний пошук лексем аж до явної передачі керування дозволяє лексичному аналізатору правильно обробляти проміжки та коментарі. Синтаксичному аналізатору лексичний аналізатор повертає єдине значення – тип лексеми [6].

У файлі правил ANTLR усі лексичні правила повинні починатись з великої літери. Ключовим словом «*fragment*» позначаються найменші неподільні лексичні правила або токени. Наведемо частину ANTLR-файлу, що реалізує функцію лексичного аналізу парсера літературних джерел:

```
DASH: '–';//'\u2013';
PYPHEN:'-';//'\u002d';
INT : [0-9]+ ;
SYMBOL: '.' | '"'| PYPHEN;
LETTER : [a-zA-Z\u0400-\u04FF];
WS : [\r\t\n]+ -> skip ;
ErrorCharacter : . ;
```

У назві ресурсу або інформації про нього можуть бути крапки або дефіси і при розборі тексту треба розпізнавати тире – як символ відокремлення однієї частини опису та іншої і дефіс – як частина опису. Ці два символи у мові Java та у правилах ANTLR виглядають однаково, але мають різні юнікоди: тире – це '\u002d', дефіс – '\u2013'.

Синтаксичний аналіз. Ієрархічний аналіз називається розбором (parsing) або синтаксичним аналізом, котрий включає групування токенів початкової програми в граматичні фрази, які використовуються компілятором (чи інтерпретатором) для синтезу виводу [7].

Ієрархічна структура програми зазвичай виражається рекурсивними правилами. У файлі правил ANTLR всі синтаксичні правила починаються з маленької

літери та описується за допомогою грама-тики у вигляді форми Бекуса-Наура (БНФ) [6]. Резуль-татом роботи ANTLR є 2 Java класи – *Lexer* та *Parser*.

Ядром ANTLR-специфікації є набір граматичних правил. Кожне правило описує синтаксичну конструкцію і дає їй ім'я. Нижче, з-за обмеженості обсягу статті, наведено частину файлу правил.

```
parse returns [List<PublicationView> publications]
@init{$publications = new LinkedList<PublicationView>();}
:
   ( INT ')' publication '.' {$publications.add($publication.publicationView);} )+
   EOF
;
publication returns [PublicationView publicationView]
@init{$publicationView = new PublicationView();}
:
 authors              {$publicationView.setАвтори($authors.authorsStr.trim());}
 sentence             {$publicationView.setНазва($sentence.text.trim());}
 ( ':' marking         {$publicationView.setПозначення($marking.text.trim());})?
 ( '/' responsible     {$publicationView.setВідповідачі($responsible.text.trim());})?
 DASH city            {$publicationView.setМісто($city.text.trim());}
 ( ':' edition         {$publicationView.setВидання($edition.text.trim());})?
 (',' INT '.'           {$publicationView.setРік($INT.text.trim());} )
 (DASH INT 'c.'       {$publicationView.setСторінок($INT.text.trim());} )
 (DASH  '(' series ';' {$publicationView.setСерія($series.text.trim());}
 'вип. №' INT ')'      {$publicationView.setВипуск($INT.text.trim());} )
;
name: sentence;
author: word ',' LETTER '.' LETTER '.' ;
city: word;
```

У кожному правилі можна використовувати звичайний Java-код, що можливо писати тільки у фігурних дужках. Також, ANTLR надає можливість оперувати його ж змінними, ставлячи знак «$» на початку змінної [2]. Завдяки цьому коду можна зрозуміти процес трансляції виразів. Головним правилом, з якого починається процес трансляції, є правило *parse*, що повертає список об'єктів PublicationView, які є відображенням таблиці з точки зору JavaFX таблиці. Правило *publication* поділено на підправила, які відповідають областям бібліографічного опису.

Таким чином, згідно з продемонстрованими правилами, парсер враховує, що у публікації можуть бути декілька авторів, назва, позначення, список відповідальних осіб або наукових заходах (конференції та ін.), місто, видання, рік, кількість сторінок, серія і випуск. Не усі області є обов'язковими, тому після деяких підправил стоїть знак «?».

Для демонстрації розглядається приклад з 4 літературними ресурсами, але у цих ресурсах заповнені не усі області. На рис. 3 добре видно вхідні параметри, коли все готово для розбору треба натиснути кнопку «Парсити», після чого таблиця у нижній половині екрану заповниться. Так як розпізнані публікації ще не збережені, то поле Id у всіх буде дорівнювати Null.

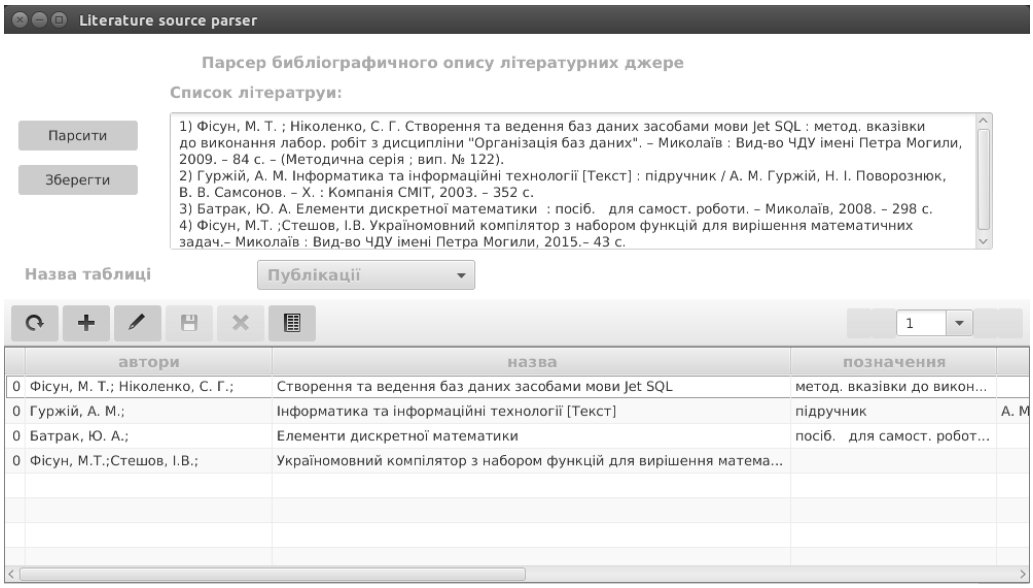

Рис. 3. Вигляд розпарсених публікацій

На рис. 4 зображено продовження таблиці, з якої помітно, що у деяких публікаціях відсутні деякі бібліографічні описи. Наприклад, ресурси 3 та 4 не мають відповідальних осіб, серії та номеру випуску, третій ресурс також немає ще й видання, а другий, хоч і не має серії та номеру, зате має перелік відповідальних осіб, на відміну від інших ресурсів.

відповідачі	місто	видання	рік	сторі...	серія	випуск
	Миколаїв	Вид-во ЧДУ імені ...	2009	84	Методична серія	122
А. М. Гуржій, Н. І. Поворознюк, В. В. Самсоно...	Х.	Компанія СМІТ	2003	352		
	Миколаїв		2008	298		
	Миколаїв	Вид-во ЧДУ імені ...	2015	43		

Рис. 4. Продовження таблиці розпарсених публікацій.

Для того, щоб оброблені джерела зберегти в базі даних, потрібно натиснути кнопку «Зберегти» і у таблиці з'явиться трохи інше відображення цих же джерел: з'явиться унікальний ідентифікатор та автори разом з видавництвами для кожної публікації будуть перелічені у стовпчик (рис. 5).

1	Фісун, М. Т.; Ніколенко, С. Г.;	Створення та ведення баз даних засобами мови Jet SQL	метод. вказівки до викон...	
2	Гуржій, А. М.;	Інформатика та інформаційні технології [Текст]	підручник	А.
3	Батрак, Ю. А.;	Елементи дискретної математики	посіб. для самост. робот...	
4	Стешов, І.В.; Фісун, М.Т.;	Україномовний компілятор з набором функцій для вирішення матема...		

Рис. 5. Вигляд збережених публікацій

Після збереження публікацій заповнюється не лише таблиця «Публікації», але й таблиці «Автори» та «Видавництва» і тепер, так як база даних не порожня, то можна продивитися інформацію у розрізі авторів та у розрізі видавництв, для чого треба вибрати відповідну таблицю у компоненті комбобокс поряд з написом «Назва таблиці». Якщо обрати таблицю з авторами (див. рис. 6), то з'явиться таблиця, у якій автори будуть унікальними, а у полі публікації може бути декілька або, якщо заповнювати базу власноруч, жодної публікації. Для зручності також додане поле «кількість публікацій», яке дозволить краще збирати статистику.

Рис. 6. Відображення таблиці авторів.

Так само, як і для таблиці авторів, таблиця з видавництвами також має поле з унікальними назвами видавництв. Програма автоматично враховує, що місто, у якому стаття опублікована, і є рідним містом видавництва. У прикладі публікації під номерами 1 та 4 були видані у видавництві ЧДУ імені Петра Могили, що програма і демонструє на рис. 7.

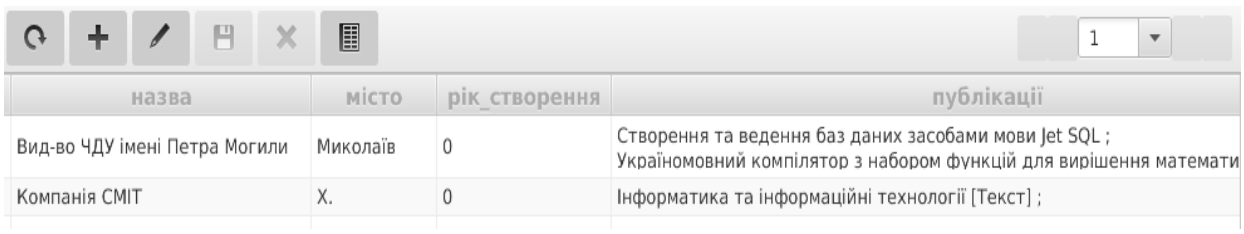

Рис. 7. Відображення таблиці видавництв

Також, для того, щоб пришвидшити пошук, публікації кожного видавництва відображаються у стовпчик. Рік створення видавництва, не належить до бібліографічного опису літературного ресурсу, тому це поле, при існуванні потреби, можна вписати власноруч, натиснувши на олівець на табличній панелі.

Усі таблиці можна редагувати та доповнювати не використовуючи парсер, так як на табличній панелі є всі необхідні інструменти. Якщо парсер робить помилку, то її легко виправити, просто змінивши запис після натиснення на іконку олівця. За додавання запису відповідає іконка з плюсом, а за видаленням – з червоним хрестиком.

Інтерфейс програми створено на основі JavaFX бібліотеки, що вбудована у JDK, починаючі з 8-ої версії. В цій бібліотеці є всі необхідні примітиви інтерфейсу сучасного графічного програмного продукту. Більш того, графічний інтерфейс можливо створювати у спеціальному *fxml* файлі, що дуже нагадує створення веб-сторінки засобами HTML. Також компоненти JavaFX підтримують CSS для стилізації. В додаток до FXML використовується бібліотека Tiwulfx, саме в якій і є графічний компонент таблиці, що має функціонал для відображення Entity об'єктів.

Висновки. Засобами компілятора компіляторів ANTLR розроблено синтаксичні правила для розбору літературних джерел відповідно до стандартів бібліографічного опису [5]. Використовуючи базу даних HSQLDB та мову Java з технологіями, що застосовуються разом з нею для спілкування з базою даних (Hibernate, Spring Data), створено засоби для зберігання літературних ресурсів та зручний графічний інструмент для заповнення та модифікування даних у базі даних.

ЛІТЕРАТУРА

1. Kosteltsev A. V. Postroeniye interpretatorov i kompiliatorov. Ispolzovaniye program BIZON, BYACC, ZUBR. – SPB : «Nauka I tehnika», 2001. – 219 c.
2. Terence Parr: Definitive ANTLR Reference. «ANTLR dlia razrabotki kompiliatora na jadzyke Java» [Elektronnyi resurs]. – Regym dostypy : URL : https://vimeo.com/25753384. – Zagol. z ekranu.
3. Steshov I. V. Ukrainomovnyi kompiliator z naborom funktsiy dlia vyrishennia matematychnyh zadach / I. V. Steshov // Vseukr. nauk.-pract. konf. molod. uchenyh, aspirantiv i studentiv «Intelectualni informatciyni systemy». – Mykolaiv : ChDU im. P. Mogyly, 2015. – C. 43–44.
4. Sistema standartov po informatsii, bibliotechnomu i izdatel'skomu delu. Bibliograficheskaya zapis'. Bibliograficheskoye opisaniye. Obshchiye trebovaniya i pravila sostavleniya [Tekst] : (GOST 7.1-2003, IDT): DSTU GOST 7.1:2006. – [Vzamen GOST 7.1-84]. – [Deystvuyushchiy s 2007-07-01]. – M. : Gospotrebstandart Ukrainy, 2007. – 47 s.
5. Publikatsii. Novyye pravila bibliograficheskogo opisaniya [Elektronnyy resurs] / A. Ustinnikova, P. Sen'ko, S. Yuldasheva [i dr.] // DSTU GOST 7.1: 2006 v Ukraine ; Gos. nauk. uchrezhdeniye «Kn. palata Ukrainy im. Ivana Fedorova». – Rezhim dostupa : URL: http://www.ukrbook.net/DSTU.htm. – Zagol. s ekrana.
6. Alfred V. Aho. Kompiliatory: printsypy, technlogii, instrumenty / Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman Aho ; Pereklad z angliiskoii. – M. : Izdatelsky dom «Williams», 2001. – 768 c.
7. Bogdanov V. L. Practychniy dustup napisannia syntaksichnogo analizatora movy programuvannia Cobol / V. L. Bogdanov, V. S. Gordeev. – 2000. – 7 c.
8. Ispol'zovaniye komp'yuternykh tekhnologiy v bibliotechno – informatsionnom obsluzhivanii predpriyatiya [Elektronnyi resurs]. – Regym dostypy : http://referat-ok.com.ua/kulturologiya-ta-mistectvo/vikoristannya-komp096yuternih-tehnologii-v-bibliotechno-informaciinomu-obslugovuvanni-pidprijemstva.
9. Karpenko I. Informatsionno – bibliotechnaya sistema «UFD / Biblioteka» // Bibliotechnyy forum Ukraina. – 2003. – № 2. – S. 15–17.
10. Dotsenko S. Integrirovannaya bibliotechnaya sistema ALEPH 500 / S. Dotsenko // Bibliotechnyy forum Ukrainy. – 2004. – № 1. – S. 11–14.
11. Voroyskiy F. S. Osnovy proyektirovaniya avtomatizirovannykh bibliotechno – informatsionnykh sistem / F. S. Voroyskiy. – Moskva : Fizmatlit, 2002.
12. UNIMARC to MARC 21 Conversion Specifications [Elektronnyi resurs]. – Regym dostypy : http://www.loc.gov/marc/unimarctomarc21.html.
13. Vikipediya: Posilannya_na_dzherela [Elektronnyi resurs]. – Regym dostypy : https://uk.wikipedia.org/wiki/Вікіпедія:Посилання_на_джерела. – Zagol. z ekranu.
14. Christopher J. Date, Vvedeniye v systemy baz dannyh / J. Christopher. – M. : Izdatelsky dom «Williams», 2001. – 1071 p.
15. Pasichnik V. V. Organizatcia baz danyh ta znan: pidruchnyk dlia VUZiv / V. V. Pasichnik, V. A. Reznichenko. – K. : BHV, 2006. – 384 c.

Фисун Н. Т., Стешов И. В.,
Черноморский государственный университет имени Петра Могилы, г. Николаев, Украина

Разработка парсера литературных источников средствами ANTLR и Hibernate для формирования библиографической базы данных

Рассматриваются правила библиографического описания литературных источников, распознаются основные части, из которых состоят источники. На основе этих частей создается структура базы данных и

связи между таблицами. Приводятся правила для реализации парсера литературных источников и инструмент выполнения правил – генератор парсеров ANTLR.

Ключевые слова: *библиографическое описание; литературный ресурс; база данных; HSQLDB; Hibernate; Spring Data; ANTLR; парсер; лексер.*

Fisun M. T., Steshov I. V.,
Petro Mohyla Black Sea State University, Mykolaiv, Ukraine

Literature resources parser developing using ANTLR and Hibernate for bibliographic database forming

In this work presented an implementation of parser that reads bibliographic description of literature resources in text format, find specific description areas and commits that in proper tables of the database. Types and rules of their bibliograohic description according to DSTU GOST 7.1:2006 were considered, specific areas that have to be recognized by parser were selected. Database structure, to what recognized and classified data is commited, was introduced. Software for lexical and syntax analyzing was developed using parser-generating tool ANTLR 4. The rules for language recognition of literature resource parser were also demonstrated. Except ANTLR for literature resource parser were used: Java 8 – basic programming language; Spring Framework 4 – databse transaction controll container; HSQLDB – database; Hibernate 4 – Java Persistence Api implementation (for ORM and communication with database); JavaFX and Tiwfulfx – library for GUI implementation; Log4j – logging library; Maven – building tool; Junit – tool for Junit testing.

A graphical user interface (GUI), that was developed using JavaFX library, that is build in JDK starting from 8 version, was described in details. This library has all necessary primitive components of modern software. JavaFX supports CSS for stilization, that is mainly used in web applications. In addition to GUI library Twilfulfx library was used in which special table graphic component exists that has functionality for displaying Entity objects. Current software is opened for further extending and improvments.

Key words: *bibliographic description; literary resource; database; HSQLDB; Hibernate; Spring Data; ANTLR; parser; lexer.*

 Дата надходження статті до редколегії 14.11.2015